Martin Engstler et al. (Hrsg.): Projektmanagement und Vorgehensmodelle 2016,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2016 121

Process Mining bei Softwareprozessen[1]

Ralf Kneuper[2][3]

Abstract: Process Mining umfasst eine Reihe von Methoden und Techniken, um Daten aus Ereignisprotokollen von Prozessen, den sogenannten Event Logs, zu analysieren und Informationen über die zu Grunde liegenden Prozesse und deren Bezug zu Prozessmodellen abzuleiten. Dieser Beitrag gibt einen Überblick der verschiedenen Möglichkeiten, Process Mining im Zusammenhang mit Softwareprozessen einzusetzen, insbesondere die Analyse des Softwareprozesses selbst, daneben aber auch die Nutzung im Rahmen des Softwareprozesses z.B. zur Anforderungsanalyse sowie die Anwendung auf die Ausführung von Softwaresystemen. Die Anwendung von Process Mining auf Softwareprozesse wird anhand einer kleinen Fallstudie ausführlicher beschrieben.

Keywords: Process Mining, Softwareprozess

1 Grundbegriffe des Process Mining

Process Mining ist eine Disziplin, die ursprünglich für Softwareprozesse entwickelt wurde (siehe Kap. 3), in den letzten Jahren aber vor allem für Geschäftsprozesse an Bedeutung gewonnen hat. Sie umfasst eine Reihe von Methoden und Techniken, um Daten aus Ereignisdaten, den sogenannten Event Logs, zu analysieren und Informationen über die zu Grunde liegenden Prozesse und deren Bezug zu Prozessmodellen abzuleiten. Es ist verwandt mit Data Mining, betrachtet aber speziell die Analyse von Prozessdaten. Diese Beschränkung des Anwendungsgebietes ermöglicht es, spezifische Methoden und Algorithmen einzusetzen sowie spezifische, prozessbezogene Fragestellungen zu untersuchen wie beispielsweise den Vergleich des durchgeführten Prozesses mit dem definierten Prozess (im Rahmen der Qualitätssicherung) oder die Analyse von Engpässen.

1.1 Nutzungsmöglichkeiten

Wie im Standardwerk [vdA11] zum Process Mining sowie dem *Process Mining Manifesto* [IEEE11] beschrieben, gibt es drei wesentliche Nutzungsmöglichkeiten dieses Ansatzes:

- Entdeckung (Discovery): Aus den verfügbaren Ereignisdaten wird der Prozess „entdeckt“ oder identifiziert. Dies ist in erster Linie relevant, wenn es (noch) keine Beschreibung oder Definition des Prozesses gibt bzw. diese unvollständig oder veraltet ist. In diesem Fall wird beispielsweise der Kontrollfluss des tatsächlich durchgeführten Prozesses oder die zu Grunde liegende Organisationsstruktur abgeleitet.

[1] Eine frühere Version dieses Beitrags ist erschienen als [Kn16].

[2] Beratung für Softwarequalitätsmanagement, Prozessverbesserung und Datenschutz, Philipp-Röth-Weg 14, 64295 Darmstadt, ralf@kneuper.de

[3] Internationale Hochschule Bad Honnef · Bonn (IUBH), Fernstudium, Campus Bad Reichenhall, Kaiserplatz 1, 83435 Bad Reichenhall, r.kneuper@iubh-fernstudium.de

- Konformität (Conformance): Der tatsächlich durchgeführte Prozess, wie er aus den Ereignisdaten erkennbar ist, wird mit dem definierten Prozess verglichen, um Abweichungen zu identifizieren. Auch in diesem Fall kann der Vergleich sich auf den Kontrollfluss des Prozesses (Werden die richtigen Schritte in der richtigen Reihenfolge durchgeführt?“) oder auf die zu Grunde liegende Organisationsstruktur („Werden die Schritte von der richtigen Rolle oder Organisationseinheit durchgeführt?“) beziehen. Hierbei handelt es sich also in erster Linie um ein Werkzeug zur Qualitätssicherung von Prozessen.
- Verbesserung (Enhancement): Aus den verfügbaren Ereignisdaten werden Prozessverbesserungspotentiale identifiziert. Diese können sich beispielsweise auf Engpässe im Prozess beziehen oder Geschäftsregeln für getroffene Entscheidungen ableiten und prüfen.

Eine eng mit Process Mining verwandte Disziplin mit großen Überschneidungen zum Process Mining ist (Business) Process Intelligence (BPI), eine Disziplin, die sich mit der Anwendung von Methoden und Techniken von Business Intelligence auf (Geschäfts-) Prozesse beschäftigt. BPI versucht, für Entscheidungen relevante Informationen aus Prozessdaten in einem Data Warehouse abzuleiten (siehe beispielsweise [CA09]). Beide Disziplinen haben große Überschneidungen und unterscheiden sich vor allem dadurch, dass BPI auch Metriken und ähnliche Informationen umfasst, andererseits bei der Analyse aber nicht unbedingt Bezug auf Prozessmodelle nimmt.

Auch wenn es mittlerweile weltweit verteilt eine Vielzahl von Gruppen gibt, die an Process Mining arbeiten und forschen, ist der Kern dieser Aktivitäten die Process Mining Group von Wil M.P. van der Aalst an der Universität Eindhoven, Niederlande, dem Autor des oben genannten Standardwerkes [vdA11].

1.2 Werkzeuge

Da mit Process Mining meist große Datenmengen auszuwerten sind, ist diese Auswertung nur mit entsprechenden Werkzeugen realistisch möglich. Das mit Abstand am weitesten verbreitete Werkzeug für diese Aufgabe ist das von der Process Mining Group der Universität Eindhoven entwickelte Open-Source-Werkzeug ProM [ProM]. Zu ProM wurden auch von vielen anderen Forschungsgruppen Plugins entwickelt, so dass ein sehr hoher Anteil aller existierenden Methoden und Techniken zum Process Mining von ProM unterstützt wird. Leider ist die Bedienung von ProM allerdings recht gewöhnungsbedürftig, und die Dokumentation, insbesondere der enthaltenen Plugins, teilweise sehr knapp.

Daneben gibt es eine Reihe weiterer Werkzeuge zur Unterstützung von Process Mining, wie die Übersicht in [IEEE11] zeigt, beispielsweise das kommerzielle Werkzeug Disco [Disco] von Fluxicon, einem Unternehmen, das von ehemaligen Mitarbeitern der Process Mining Group der Universität Eindhoven gegründet wurde. Disco ist nicht ganz so mächtig wie ProM, dafür aber gerade für Einsteiger sehr viel einfacher zu bedienen.

1.3 Ereignisprotokolle

Grundlage des Process Minings sind Ereignisprotokolle, die Informationen zu den Ereignissen enthalten, die bei der Ausführung eines Prozesses aufgetreten sind. Gemäß [vdA11, § 4.2] basiert ein Ereignisprotokoll auf folgenden Annahmen:

- Ein Prozess (bzw. seine Ausführung) besteht aus Fällen, beispielsweise kann der Prozess der Softwareentwicklung in einzelne Anforderungen gegliedert werden. Ein Fall besteht dann aus der Bearbeitung einer solchen Anforderung.
- Ein Fall besteht aus verschiedenen Ereignissen, die jeweils genau einem Fall zuzuordnen sind. Bei der Softwareentwicklung ist ein Ereignis dann beispielsweise die Analyse, Genehmigung oder Implementierung einer Anforderung.
- Ereignisse, die zu einem Fall gehören, sind geordnet. Aus dieser Ordnung lässt sich dann der im Prozess bestehende Kontrollfluss ableiten. Diese Ordnung der Ereignisse wird häufig über Zeitstempel hergestellt, wodurch auch eine gemeinsame Ordnung über Ereignisse zu einem Fall möglich wird, wenn diese in verschiedenen Werkzeugen (z.B. einem Ticketsystem und einem Konfigurationsmanagementsystem) liegen.
- Ereignisse können Attribute haben wie die dem Ereignis zugeordnete Aktivität, Zeitpunkt, Kosten oder beteiligte Ressource(n).

Daraus ergibt sich, dass ein für Process Mining zu verwendendes Ereignisprotokoll mindestens die relevanten Fälle sowie die zugeordneten Aktivitäten (Ereignisse) und deren Reihenfolge protokollieren muss, für weiterführende Analysen auch entsprechend weitere Attribute.

Um derartige Ereignisprotokolle mit verschiedenen Werkzeugen generieren und bearbeiten zu können, wurde als Standardformat zuerst die *Mining eXtensible Markup Language* (MXML) [MXML] definiert, welche mittlerweile durch das Format *eXtensible Event Stream* (XES) abgelöst wurde, siehe [vdA11, § 4.3] und [XES].

Ein Beispiel für eine solche XES-Datei ist in Listing 2 zu finden.

Das Werkzeug ProM hat einige Importfunktionen, um aus anderen Formaten derartige XES-Protokolle generieren zu können. Ein weiteres einfaches, XLS2XES genanntes Konvertierungswerkzeug, um aus Excel-Tabellen XES-Ereignisprotokolle zu erstellen, wurde vom Autor entwickelt und ist verfügbar unter `http://kneuper.de/ProcessMining/`.

2 Process Mining und Softwareentwicklungsprozesse

Obwohl die ersten Veröffentlichungen zu Process Mining (wenn auch noch nicht unter dieser Bezeichnung) sich auf Softwareprozesse bezogen, siehe [CW95, CW98], wurde Process Mining seither in erster Linie auf Geschäftsprozesse angewendet. Die Anwendung der Ideen und Konzepte von Process Mining auf das IT-Servicemanagement ist relativ

einfach möglich, da es sich hier meist um strukturierte, wiederholbare und häufig wiederholte Prozesse handelt. Aus diesem Grund bezog sich auch der *Business Process Intelligence Challenge* in den Jahren 2013 und 2014 auf IT-Servicemanagementprozesse [BPIC13, BPIC14]. Im IT-Servicemanagement kann Process Mining beispielsweise genutzt werden, um die Einhaltung von Service Level Agreements (SLA) zu überprüfen bzw. die Ursachen bei Nicht-Einhaltung zu analysieren. Hier gibt es gewisse Überschneidungen mit klassischen Kennzahlensystemen für das IT-Servicemanagement, aber durch den stärkeren Bezug auf den gesamten Prozess statt einzelner Prozessschritte geht das Process Mining über solche Kennzahlensysteme hinaus.

Softwareentwicklungsprozesse sind dagegen meist weniger strukturiert und werden seltener durchgeführt (oder — anders formuliert — die einzelne Durchführung des Prozesses bzw. seiner Aktivitäten dauert typischerweise wesentlich länger), so dass die Anwendung von Process Mining hier entsprechend schwieriger ist. Trotzdem gibt es einige Ansätze zur Nutzung von Process Mining im Bereich der Softwareentwicklung:

- Anwendung auf die Softwareentwicklungsprozesse selbst: dies ist die offensichtlichste und direkteste Form, siehe Abschnitt 3 unten.
- Anwendung auf die durch IT unterstützten bzw. zu unterstützenden Anwendungsprozesse, beispielsweise zur Analyse von Anforderungen: Dieser Ansatz wird in Abschnitt 5 ausführlicher behandelt.
- Anwendung auf den ausgeführten Programmcode als Prozess, also eine eher technische Betrachtung. Dieser Ansatz dient beispielsweise zur Analyse und Steigerung der Performanz der Software, siehe Abschnitt 6.

Schwerpunkt dieses Beitrags ist der erstgenannte Ansatz, der daher im Weiteren relativ ausführlich behandelt wird, während zu den beiden anderen Ansätzen nur eine kurze Übersicht gegeben wird.

Ebenfalls eng verwandt mit dem Process Mining, auch wenn es sich nicht nur auf Prozesse bezieht, ist das Mining von Software Repositories (MSR). Dieses Thema wird u.a. in einer Reihe von Teilkonferenzen der ICSE-Konferenzen (siehe http://msrconf.org) behandelt, mit einem breiten Spektrum von Fragen. Dazu gehören beispielsweise Fragen zur Arbeitsweise von Entwicklern (z.B. „Welche Arten von Fragen stellen Entwickler in relevanten Foren? Wie kooperieren Entwickler in Projekten?“), zu Produkten und deren Qualität (z.B. „Welche Arten von Fehlern und Warnungen treten häufig auf?“) und schließlich auch zu den hier behandelten Prozessen.

3 Mining von Softwareentwicklungsprozessen

Die ersten Veröffentlichungen zu Process Mining bei Softwareentwicklungsprozessen waren [CW95, CW98] und beschrieben drei verschiedene Methoden zur Entdeckung von Prozessen auf Basis von algorithmischer Inferenz von Grammatiken, von Markov-Modellen und von neuronalen Netzwerken.

Während die Algorithmen und Werkzeuge zum Process Mining bei Softwareentwicklungsprozessen im Wesentlichen die gleichen sind wie für allgemeine Geschäftsprozesse,

sind die Datenquellen sowie die Vorbereitung der Daten deutlich anders. Bei Softwareentwicklungsprozessen sind die Daten meist in einem Repository enthalten, beispielsweise in Konfigurationsmanagement-Werkzeugen wie Git oder Subversion, oder Werkzeugen für Anforderungsmanagement und Bug Tracking, beispielsweise BugZilla oder Flyspray. Ereignisse, die im Process Mining betrachtet werden können, sind dann beispielsweise ein Commit im Konfigurationsmanagement-Werkzeug, eine Änderung des Status eines Tickets im Bug Tracking System, oder die Zuweisung eines solchen Tickets an einen anderen Verantwortlichen. Je nachdem, welche Informationen mit Process Mining gewonnen werden sollen, müssen die auszuwertenden Ereignisse passend selektiert werden.

Sehr relevant, aber meist wenig strukturiert und daher schwierig zu analysieren sind Mail-Datenbanken oder Diskussionsforen innerhalb von Projekten. Die Analyse dieser Form von Daten wird beispielsweise in der oben angesprochenen MSR-Konferenzreihe behandelt.

Wenn die Daten dann als Ereignisprotokoll aufbereitet sind, können sie ähnlich wie allgemeine Geschäftsprozesse mit den üblichen Methoden des Process Mining analysiert werden.

Die Analyse von Repositories im Software Engineering, um daraus für das Process Mining geeignete Daten bzw. Ereignisprotokolle zu gewinnen, wurde insbesondere von Poncin in seiner Masterarbeit [Po10] und den weiteren Publikationen [PSv11b, PSv11a] betrachtet. Als Teil dieser Arbeit entwickelte Poncin das *FRamework for Analyzing Software Repositories* (FRASR), mit dem die Extraktion von Ereignisdaten aus verbreiteten Repositories wie Subversion und Git unterstützt wird.

Auf Basis dieser Vorbereitung von Ereignisdaten untersuchte Poncin dann beispielhaft die Zuordnung von Projektrollen zu Einzelpersonen, also die Frage, inwieweit man aus den Ereignisdaten ableiten kann, welche Person welche Prozessrolle innehat (beispielsweise bei einem Open Source-Projekt). Als zweites Beispiel untersuchte er, inwieweit die tatsächliche Bearbeitung von Fehlermeldungen einem definierten Lebenszyklus entspricht.

Eine ähnliche Anwendung von Process Mining beschreibt [GTK14]. Hier wird Process Mining als Werkzeug zur Prüfung von Softwareentwicklungsprozessen beschrieben mit den folgenden drei Komponenten:

- *Process Variant Miner*: Diese Komponente untersucht, welche Prozessvarianten in der Entwicklung tatsächlich umgesetzt wurden.
- *Conformance Analyser*: Diese Komponente vergleicht die Umsetzung des Prozesses gegen seine Spezifikation zur Überprüfung der Konformität.
- *Statistical Analyser*: Aufbauend auf den Ergebnissen der ersten beiden Komponenten wertet diese Komponente die Ergebnisse statistisch aus, um Aussagen über die Stabilität des Prozesses und seiner Varianten zu liefern.

[SK14] beschreibt ebenfalls die Nutzung von Process Mining zur Konformitätsprüfung, in diesem Fall als Bestandteil von CMMI-Appraisals.

Sowohl [Po10] also auch [GTK14] enthalten zusätzlich einen guten Überblick über relevante Literatur zum Process Mining von Softwareentwicklungsprozessen, so dass sie eine hilfreiche Basis für eine tiefergehende Einarbeitung in dieses Thema bilden.

Auffällig ist, dass in allen drei Arbeiten [Po10, GTK14, SK14] fast die gleichen Prozesse als Beispiel verwendet werden, nämlich die Bearbeitung von Fehlermeldungen und Änderungsanträgen. Diese Prozesse sind, im Gegensatz zu anderen Softwareentwicklungsprozessen, meist relativ gut strukturiert, mit häufigen Wiederholungen der einzelnen Prozessschritte, und Process Mining lässt sich daher relativ gut anwenden.

Ähnliches gilt bei iterativem Vorgehen mit kurzen Iterationen für den Prozess der Bearbeitung einer einzelnen Anforderung oder Story mit Analyse der Aufgabenstellung, Testfalldefinition, Design und Implementierung, Testdurchführung (meist in mehreren Stufen), und Commit der Ergebnisse. Auch hier ist Process Mining anwendbar, um beispielsweise Aussagen über die Konformität zum vereinbarten Vorgehen oder über Prozessschritte, bei denen häufig Verzögerungen auftreten, zu machen.

Bei den meisten anderen Softwareentwicklungsprozessen gilt das nicht und die Anwendung von Process Mining ist wesentlich schwieriger bzw. die Ergebnisse sind weniger aussagekräftig.

Auch bei der Anwendung von Process Mining auf die Bearbeitung von Fehlermeldungen und Änderungsanträgen sind sinnvolle Aussagen über die Konformität aber oft schwierig: Häufig wird hierfür ein Ticketsystem verwendet, das von vornherein nur erlaubte Zustandsübergänge zulässt, bzw. es sind sehr viele oder sogar alle grundsätzlich möglichen Übergänge auch erlaubt. In diesem Fall kann man mit Process Mining natürlich keine Abweichungen finden, ausgenommen solche, die durch eine frühere andere Konfiguration entstanden sind. Hilfreich kann Process Mining dagegen bei Abweichungen sein, die durch komplexere, nicht im Ticket-System modellierte Abläufen entstehen, wie z.B. einem Bearbeitungs- oder Genehmigungsschritt, der nur bei Tickets ab einem bestimmten Schätzaufwand gefordert wird, ohne dass diese Bedingung im Ticketsystem modelliert werden konnte.

4 Fallstudie zum Process Mining von Softwareentwicklungsprozessen

Am Beispiel mehrerer kleiner Entwicklungsprojekte, die mit der gleichen Instanz des Bug-Tracking-Systems *Flyspray* [Fly] verwaltet werden, sollen die beschriebenen Konzepte und die allgemeine Vorgehensweise kurz dargestellt werden.

Erster Schritt zum Process Mining ist die Erzeugung eines Ereignisprotokolls:

- Als *Prozess* wird die Bearbeitung von Tickets in Flyspray betrachtet.
- Ein *Fall* ist dann die Bearbeitung eines bestimmten Tickets.
- *Ereignisse* sind die einzelnen in Flyspray festgehaltenen Bearbeitungsschritte.

Eine Analyse der von Flyspray verwendeten Datenbank und der entsprechenden Datenstrukturen zeigt, dass die wesentlichen zum Process Mining benötigten Informationen in der Tabelle `flyspray history` enthalten sind. Mit einem entsprechenden SQL-Skript (siehe Abb. 1) können daraus die wichtigsten Daten extrahiert werden. Dabei sind folgende Aspekte zu berücksichtigen:

- Obwohl die Flyspray-Instanz und damit auch die Tabelle `flyspray history` mehrere Projekte enthält, werden diese hier nicht separat betrachtet, da allen das gleiche Vorgehen und die gleichen Zustände und Zustandsübergänge zugrunde liegen.[4]
- Fokus der Analyse soll in diesem Fall die Betrachtung der Folge von Zuständen sein. Daher werden die Zustandsübergänge als Ereignisse interpretiert, mit jeweils dem neuen Zustand als durchgeführte „Aktivität". Außerdem werden nur die Aktivitäten als Ereignisse übernommen, die zu einem Zustandswechsel führen, während beispielsweise ein Wechsel des Verantwortlichen oder eine Anpassung der Beschreibung hier nicht weiter betrachtet wird:
  - `event_type=1`: Aufgabe angelegt
  - `event_type=2`: Aufgabe geschlossen
  - `event_type=3`: Feld geändert, wobei nur eine Änderung des Feldes `item_status` relevant ist, und in diesem Fall der neue Status (als Zahlencode) ausgelesen und über einen `JOIN` in den zugehörigen Textwert übersetzt werden muss.
  - `event type=13`: Aufgabe wieder geöffnet
- Aus dem gleichen Grund wurden andere vorhandene Informationen wie z.B. die „Ressource", also die für eine Aktivität verantwortliche Person oder Rolle, oder die Zuordnung eines Tickets zu einer bestimmten Kategorie, nicht in das Ereignisprotokoll übernommen.

[4] Natürlich ist es auch möglich, die verschiedenen Projekte separat zu betrachten. Da die dafür benötigte project id allerdings nicht in flyspray history abgespeichtert ist, wäre dafür ein weiterer JOIN mit der Tabelle flyspray tasks erforderlich.

```
SELECT `history_id`, `task_id`, `event_date`, "offen"
FROM flyspray_history
WHERE event_type=1
UNION
SELECT `history_id`, `task_id`, `event_date`, "closed"
FROM flyspray_history
WHERE event_type=2
UNION
SELECT `history_id`, `task_id`, `event_date`,
fls_new.status_name AS new_status
FROM `flyspray_history`
JOIN flyspray_list_status AS fls_new
ON flyspray_history.new_value=fls_new.status_id
WHERE field_changed='item_status' AND event_type=3
UNION
SELECT `history_id`, `task_id`, `event_date`, "offen"
FROM flyspray_history
WHERE event_type=13
ORDER BY task_id,event_date
```

Abb. 1: SQL-Code für Ereignisprotokoll aus Flyspray

Im nächsten Schritt wurden die Ergebnisse als CSV-Datei exportiert und mit dem bereits angesprochenen Werkzeug XLS2XES in ein XES-Ereignisprotokoll konvertiert. Ein Ausschnitt aus dieser Datei ist in Listing 2 enthalten.

```
<? xml version =" 1.0 " encoding ="UTF -8" ?>
<! -- XES version 1.0 -->
<! -- Created by Excel macro XLS2XES -->
<! -- Created 2016 -05 -24 17 :10:54 -->
<! -- Columns:  Case ID: b;  Resource: ( none ); Timestamp:
c;  Activity: e;  Role ID: ( none );
Life cycle transition: ( none ) -->
<log xes . version =" 1.0 " xes . features =" nested -
attributes " xmlns =" http: // www .xes - standard . org
"
xes . creator =" XLS2XES ">
< extension name =" Lifecycle " prefix =" lifecycle "
uri =" http: // www .xes - standard . org / lifecycle .
xesext " />
< extension name =" Organizational " prefix =" org "
uri =" http: // www .xes - standard . org / org . xesext
" />
...
<string key =" concept:name "
value =" flyspray_history_20160524 " />
<string key =" lifecycle:model " value =" standard " />
<trace >
<string key =" concept:name " value ="1" />
<event >
<date key =" time:timestamp "
value =" 2005 -10 -22 T23:46:37 " />
<string key =" concept:name " value =" offen " />
</ event >
<event >
<date key =" time:timestamp "
value =" 2010 -12 -18 T20:22:31 " />
<string key =" concept:name " value =" closed " />
</ event >
...
</ trace >
<trace >
...
</ trace >
...
</ log >
```

Abb. 2: Beispiel-Ereignisprotokoll im XES-Format

Mit verschiedenen Process-Mining-Algorithmen kann man dieses Ereignisprotokoll nun analysieren und erhält dadurch beispielsweise ein Prozessdiagramm[5] (siehe Abb. 3) oder eine Auswertung der Dauer der einzelnen Fälle (siehe Abb. 4).

[5] Hierbei ist zu beachten, dass dieses Diagramm die Aktivitäten im Prozess beschreibt, d.h. ein Wert wie „in Arbeit“ beschreibt nicht den gleichnamigen Zustand, sondern den Übergang in diesen Zustand.

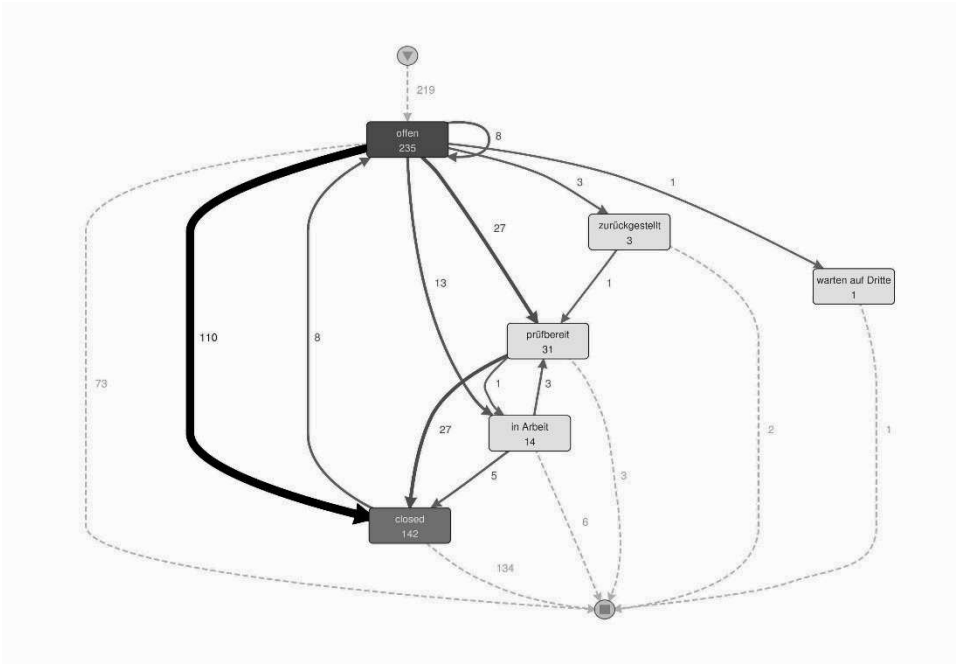

Abb. 3: Prozessdarstellung

Eine erste Analyse dieser Ergebnisse zeigt, dass viele Fälle (Tickets) nicht wie erwartet auf „prüfbereit“ gesetzt waren, bevor sie geschlossen wurden. Im Gegenteil wurde sogar etwa die Hälfte aller Tickets direkt vom Zustand „offen“ in den Zustand „closed“ [6] überführt.

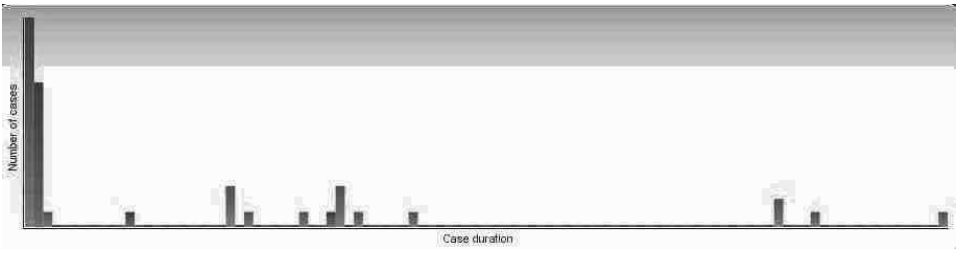

Abb. 4: Falldauer

Hier wurde daher eine Klärung notwendig, ob und in welchen Fällen der explizite Wechsel in den Zustand „prüfbereit“ tatsächlich“ erwartet wird.

Eine explizite Konformitätsbewertung ist in diesem Beispiel allerdings nicht angemessen, da die verwendeten Zustände zwar implizit eine Reihenfolge vorgeben, es aber keinen definierten und einzuhaltenden Bearbeitungsprozess gibt.

[6] Das unschöne Durcheinander von deutschen und englischen Zustandsbezeichnungen ist dadurch entstanden, dass Standardbezeichnungen des Systems nicht durchgängig angepasst wurden, neue Zustandsbezeichnungen aber immer in deutscher Sprache eingetragen wurde.

Aus Abb. 4 ist zu erkennen, dass die Falldauer sehr variiert (auch wenn in der Abbildung die Wertebereiche nicht erkennbar sind), wobei ein großer Teil der Tickets sehr schnell erledigt wurde, beim Rest die Bearbeitungsdauer aber sehr unterschiedlich war. Das ist in diesem Beispiel nicht anders zu erwarten, da Priorität und Aufwand der zu bearbeitenden Tickets ebenfalls sehr unterschiedlich sind. Im nächsten Schritt wurde darauf aufbauend geprüft, wie lange die Tickets in den jeweiligen Zuständen verbleiben. Während das für die meisten Zustände nicht relevant ist, sollten Tickets aber nicht lange in einem der Zustände „in Arbeit“ oder „prüfbereit“ sein. Bis auf einzelne Ausnahmen konnte das bestätigt werden.

5 Mining der Anwendungsprozesse

Eine Alternative ist die Anwendung von Process Mining nicht auf Softwareentwicklungsprozesse, sondern als ein Schritt in einem solchen Prozess, beispielsweise um Anforderungen zu identifizieren und zu analysieren. Dabei kann Process Mining einerseits auf einen Prozess angewendet werden, dessen IT-Unterstützung erst noch entwickelt werden soll, in erster Linie also zur Analyse der Anforderungen an das zu entwickelnde System [MF08, RLv14, RM14], andererseits zur Analyse der Nutzung eines bereits bestehenden IT-Systems.

Im Wesentlichen gibt es hierfür folgende Anwendungsszenarien:

- Wenn ein neues System entwickelt werden soll, dann werden durch Process Mining die bisher genutzten Geschäftsprozesse analysiert, um zu verstehen, was die Benutzer tun und welche Schritte das neue System unterstützen könnte und sollte, beispielsweise um häufige Fehler zu vermeiden oder lange dauernde Prozesse zu beschleunigen.
- Wenn entsprechende Software bereits existiert und eingesetzt wird, dann kann Process Mining von Nutzungsprotokollen dabei helfen, die Nutzung dieser Software besser zu verstehen und dadurch die Software zu verbessern. Wie interagieren die Benutzer mit dem System, welche Schritte führen sie dabei aus, und welche typischen Probleme treten dadurch auf, sei es durch Fehlbedienung oder weil erwartete Funktionen nicht vorhanden sind?

  Bei einer iterativen Vorgehensweise kann dieses zweite Szenario in gewissem Rahmen auch bei der Entwicklung neuer Software genutzt werden, da man hier schon sehr früh existierende Software hat, die testweise oder sogar produktiv genutzt werden kann. Auch hier wird die Nutzung dieser neuen Softwareversionen, beispielsweise am Ende eines Scrum-Sprints, protokolliert und das resultierende Protokoll dann für die Planung der folgenden Sprints durch Methoden des Process Mining ausgewertet werden. Diese Vorgehensweise ist beispielsweise in [RM14] ausführlicher beschrieben.
- Eng damit verwandt ist die in [vM14] beschriebene Anwendung von Process Mining auf die Nutzung einer neuen Softwareversion, um diese gegen die Altversion zu vergleichen als Basis für eine Entscheidung für oder gegen die Einführung der

neuen Version. Dazu werden die Ereignisprotokolle für ausgewählte Standardabläufe mit beiden Versionen mit Hilfe von Process Mining verglichen, z.B. auf Bearbeitungsgeschwindigkeit oder Fehlerhäufigkeit.

- Ein weiteres Szenario ist die Nutzung von Process Mining, um im Rahmen des Software Reengineerings die Funktion eines Altsystems zu rekonstruieren [LMP08, FHR11, PW11].

6 Process Mining der Ausführung des Programmcodes

Bei dieser dritten Variante betrachtet man den Programmcode als Prozess, der ausgeführt und dann mit Process Mining analysiert wird. Voraussetzung dafür ist u.a., dass das System entsprechend instrumentiert wurde, um seine Ausführung in einem Event Log zu protokollieren.

Dieser eher selten verwendete Ansatz kann beispielsweise dazu dienen, Engpässe zu identifizieren und die Performanz eines Systems zu analysieren und zu steigern.

7 Ausblick

Während Process Mining für die Prozesse des IT-Servicemanagements wie beispielsweise Incident Management oder Change Management problemlos anwendbar ist und in Unternehmen gelegentlich auch bereits angewendet wird, gilt dies für Softwareentwicklungsprozesse in geringerem Umfang. Auch hier ist Process Mining allerdings zumindest auf manche Prozesse der Softwareentwicklung anwendbar, je nach Grad der Strukturierung des Prozesses mehr oder weniger leicht. Tendenziell gilt das vor allem bei agilem oder iterativem Vorgehen, da hier die einzelnen Arbeitsschritte meist kleiner und in den entsprechenden Werkzeugen besser nachvollziehbar sind.

Daneben kann Process Mining auch als ein Schritt innerhalb der Softwareentwicklung eingesetzt werden, um die zu unterstützenden Prozesse bzw. die Nutzung des entwickelten Systems zu analysieren, oder um bestehende Softwaresysteme und deren Ausführung zu analysieren.

Offene Fragen für die weitere Forschung in diesem Feld sind daher beispielsweise:

- Für welche Teilprozesse der Softwareentwicklung ist Process Mining geeignet, für welche eher nicht?
- Welche nützlichen Informationen können realistisch durch Process Mining abgeleitet werden?
- Inwieweit ist Process Mining beispielsweise ein gutes Werkzeug für die Qualitätssicherung von Softwareprozessen?
- Wie kann man Daten aus verschiedenen Systemen integrieren und gemeinsam auswerten, z.B. aus Konfigurationsmanagementwerkzeugen und Fehlerverwaltungssystemen?

- In welchen Fällen ist Process Mining ein geeignetes Werkzeug zur Analyse von Anforderungen, und wie kann es in diesen Fällen am besten in die Softwareentwicklung integriert werden?

Literaturverzeichnis

[BPIC13] 9th International Business Process Intelligence Challenge (BPIC'13), http://www.win.tue.nl/bpi/2013/challenge.

[BPIC14] 10th International Business Process Intelligence Challenge (BPIC'14), http://www.win.tue.nl/bpi/2014/challenge.

[CA09] Castellanos, M.; Alves De Medeiros, A.K.; Mendling, J.; Weber, B.; Weijters, A.J.M.M.: Business process intelligence. Information Science Reference, Hersey, S. 456–480, 2009.

[CW95] Cook, Jonathan E.; Wolf, Alexander L.: Automating Process Discovery through Event-Data Analysis. In: Proc. of the 17th Intern. Conf. on Software Engineering, Seattle, Washington, USA. S. 73–82, 1995.

[CW98] Cook, Jonathan E.; Wolf, Alexander L.: Discovering Models of Software Processes from Event-Based Data. ACM Transactions on Software Engineering and Methodology, 7(3):215–249, 1998.

[Disco] Disco—Discover your processes, http://fluxicon.com/disco/.

[FHR11] Fuhr, Andreas; Horn, Tassilo; Riediger, Volker: Using Dynamic Analysis and Clustering for Implementing Services by Reusing Legacy Code. In (Martin Pinzger, Denys Poshyvanyk; Buckley, Jim, Hrsg.): 18th Working Conference on Reverse Engineering. IEEE Computer Society, S. 275–279, 2011.

[Fly] Flyspray. The bug killer!, http://www.flyspray.org/.

[GTK14] Gürgen, Tugba; Tarhan, Ayca; Karagöz, N. Alpay: An Integrated Infrastructure Using Process Mining Techniques for Software Process Verification. In (Perez-Castillo, R.; Piattini, M., Hrsg.): Uncovering Essential Software Artifacts through Business Process Archeology, Kapitel 14, S. 364–382. IGI Global, 2014.

[IEEE11] IEEE Task Force on Process Mining: Process Mining Manifesto. http://www.win.tue.nl/ieeetfpm/doku.php?id=shared:process_mining_manifesto, 2011.

[Kn16] Kneuper, Ralf: Process Mining bei Softwareprozessen: Ein Überblick. Softwaretechnik-Trends, 36(1):16–19, 2016.

[LMP08] Lorenzoli, Davide; Mariani, Leonardo; Pezzè, Mauro: Automatic Generation of Software Behavioral Models. In: Proc. of the 30th International Conference on Software Engineering (ICSE '08). ACM, S. 501–510, 2008.

[MF08] Maruster, Laura; Faber, Niels; Jorna, Ren J.; van Haren, Rob J. F.: A process mining approach to analyse user behaviour. In: WEBIST 2008, Proceedings of the Fourth International Conference on Web Information Systems and Technologies, Funchal, Madeira, Portugal. Jg. 2. INSTICC Press, S. 208–214, 2008.

[MXML] MXML (Mining eXtensible Markup Language), http://www.processmining.org/logs/mxml.

[PW11] Pérez-Castillo, Ricardo; Weber, Barbara; García-Rodríguez de Guzmín, Ignacio; Piattini, Mario: Process mining through dynamic analysis for modernising legacy systems. IET Software, 5(3):304–319, 2011.

[Po10] Poncin, Wouter: Process Mining Software Repositories. Masterarbeit, Eindhoven University of Technology, August 2010. Verfügbar unter http://www.frasr.org/ downloads/2010-08-20_ThesisWouterPoncin.pdf.

[ProM] ProM Tools, http://www.promtools.org/doku.php.

[PSv11a] Poncin, Wouter; Serebrenik, Alexander; van den Brand, Mark: Mining student capstone projects with FRASR and ProM. In: SPLASH 2011 Educators' Symposium, Portland OR, USA. 2011. Verfügbar unter http://www.frasr.org/downloads/2011-10_ SPLASH-ETS.pdf.

[PSv11b] Poncin, Wouter; Serebrenik, Alexander; van den Brand, Mark: Process mining software repositories. In: 15th European Conference on Software Maintenance and Reengineering, March 1-4, 2011, Oldenburg, Germany. 2011. Verfügbar unter http://www.frasr. org/downloads/2011-03_CSMR.pdf.

[RLv14] Rubin, Vladimir A.; Lomazova, Irina A.; van der Aalst, Wil M.P.: Agile Development with Software Process Mining. In: Proceedings of the 2014 International Conference on Software and System Process (ICSSP 2014). ACM, S. 70–74, 2014.

[RM14] Rubin, Vladimir A.; Mitsyuk, Alexey A.; Lomazova, Irina A.; van der Aalst, Wil M. P.: Process Mining Can Be Applied to Software Too! In: Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM'14. ACM, S. 57:1–57:8, 2014.

[SK14] Samalikova, J.; Kusters, Rob J.; Trienekens, Jos; Weijters, A.J.M.M.: Process mining support for Capability Maturity Model Integration-based software process assessment, in principle and in practice. Journal of Software: Evolution and Process, 26(7), 2014.

[vdA11] van der Aalst, Wil M.P.: Process Mining. Discovery, Conformance and Enhancement of Business Processes. Springer, 2011.

[vM14] van Genuchten, Michiel; Mans, Ronny; Reijers, Hajo; Wismeijer, Daniel: Is Your Upgrade Worth It? Process Mining Can Tell. IEEE Software, 31(5):94–100, 2014.

[XES] XES Extensible Event Stream, www.xes-standard.org.